REMARKS/ARGUMENTS

The Applicant has filed the present Amendment pursuant to 37 C.F.R. § 1.116 in reply to the outstanding Final Rejection of October 27, 2003. The Applicant believes the present Amendment to be fully responsive to the Final Rejection for reasons set forth below.

In the Final Rejection, the Examiner has maintained the rejection of Claims 27-47 pursuant to 35 U.S.C. § 101, as allegedly directed to non-statutory subject matter. The Examiner has further rejected Claims 27-32 and 37-47 pursuant to 35 U.S.C. § 102(b), as allegedly anticipated by Wold (U.S. Patent No. 5,724,589).

Regarding the non-statutory subject matter rejection of Claims 27-47 pursuant to 35 U.S.C. § 101, the Applicant has amended the independent Claims 27 and 47 as requested by the Examiner, to explicitly recite that the software unit is stored on a computer readable medium and executable by a computer in a software system that includes a plurality of software units stored on the computer readable medium. Therefore, the Applicant respectfully submits that the new Claims 27-47 are statutory subject matter in accordance with the strictures pursuant to 35 U.S.C. § 101. The Applicant further respectfully submits that no new subject matter has been entered. Consequently, the Applicant respectfully requests the Examiner to withdraw the rejection of Claims 27-47 pursuant to 35 U.S.C. § 101.

The Applicant respectfully disagrees with the Examiner's allegations regarding the rejection of the independent Claims 27 and 47 pursuant to 35 U.S.C.

§ 102(b), and as a consequence, the Applicant proffers the following distinguishing arguments to traverse the rejection over the prior art reference to Wold.

In traversing the rejection of the independent Claims 27 and 47 pursuant to 35 U.S.C. § 102(b), the Applicant respectfully submits that Wold is defective in that it fails to disclose a software unit which comprises a variable accessed at runtime of the software unit that identifies the one or more of the plurality of software units that is to receive the message and identifies the method that is to be executed at the identified one or more of the plurality of software units. Wold is directed to a development system providing a programming event model for developing context-free reusable software components. More specifically, Wold discloses a concept of "closure" for providing reusability of software components, i.e., the notion of binding or connecting "where to call" and who is being called." (See Wold Col. 4, lines 8-19). As particularly depicted in Wold's Fig. 5, Wold discloses a double pointer to a static function 531 in a calling object of Class B 530 to obtain both an address for a static dispatcher member function 517 and an address of a called object of Class A 510. Wold's further discloses that the static dispatcher function 517 then calls a virtual function in the called object 510 as depicted in Wold's Fig. 5, i.e. the static dispatcher function 517 "dispatches" a call onto the actual virtual function in the called object 510 (See Wold Co. 8, lines 23-27). Still further, Wold discloses using the double pointer to a static function also as a parameter for enabling Wold's system to calculate the address of the called object of Class A 510, i.e., "this" pointer for called object of Class A (See Wold Col. 9, lines 1-14).

In the rejection of the independent Claims 27 and 47 pursuant to 35 U.S.C. § 102(b), the Examiner alleged that Wold's static dispatcher member function 517 (See

Wold Col. 4, lines 33-41) discloses a variable accessed at runtime of the software unit that identifies the one or more of the plurality of software units that is to receive the message and identifies the method that is to be executed at the identified one or more of the plurality of software units, as particularly recited in the independent Claims 27 and 47. To the contrary of the Examiner's allegation, the Applicant respectfully submits that Wold's static dispatcher member function (Wold Col. 4, lines 33-41; Fig. 5, no. 517) does not disclose the claimed variable as recited in the independent Claims 27 and 47. First, unlike the claimed invention's variable, Wold's static dispatcher member function 517 is not a variable, but it is indeed a function invoked to further call a virtual member function. Second, unlike the claimed invention's variable, the static dispatcher member function is not a component of the calling object of Class B 530. That is, the static dispatcher member function is a function in the called object of Class A 510, not the calling object of Class B 530. However, to the contrary of Wold, the recited variable is a component of the software unit that identifies the one or more of the plurality of software units that is to receive the message and identifies the method that is to be executed at the identified one or more of the plurality of software units. Consequently, the Applicant respectfully submits that to the contrary of the claimed invention, Wold's static dispatcher member function 517 fails to disclose the claimed variable.

Further regarding the rejection of the independent Claims 27 and 47 pursuant to 35 U.S.C. § 102(b), the Applicant further respectfully submits that Wold's double pointer to a static function 531 in the calling object of Class B 530 also does not disclose the claimed variable as particularly recited in Claims 27 and 47. That is, Wold does not disclose the claimed variable that identifies one or more of the plurality of

software units that is to receive the message. More specifically, unlike the claimed variable, Wold's double pointer to static function 531 does not directly identify the called object of class A 510. More specifically, the double pointer to the static dispatcher member function is passed as an argument and the Wold's system calculates the address of the called object's "this" pointer to derive the object's address (See Wold Col. 9, lines 1-14). However, to the contrary of Wold, the claimed software unit's variable identifies the one or more software units that is to receive the message, i.e., the variable identifies the addresses of the one or more software units. More specifically, Wold's calling object of Class B 530 does not know the address of the of the called object of Class A 510, but only knows the address of the static dispatcher member function 517 within object of Class B 530. In fact, Wold teaches away from the claimed variable because Wold discloses that it is desired that its calling object of Class B 530 need not know the details of the called object of Class A 510, and vice versa (See Wold Col. 8, lines 43-44).

Still further regarding the rejection of the independent Claims 27 and 47 pursuant to 35 U.S.C. § 102(b), Wold does not disclose the claimed variable that identifies the method at the one or more identified software units that is to be executed, i.e., the address of the method to be executed. More specifically, Wold uses the static dispatcher function 517 as described hereinabove to invoke or dispatch a call to the virtual function within the called object of Class A 510. More specifically, to the contrary of the claimed variable that identifies the method which is to be executed at the at the identified one or more software unit, Wold's calling object of Class B 530 does not know the address of the virtual member function that is to be invoked in the called object of Class A 510. That is, Wold invokes the static dispatcher function 517 in the called object of Class A 510 as described hereinabove, which in turns actually invokes the

desired virtual function within the called object of Class A 510. Consequently, the Applicant respectfully submits that to the contrary of the claimed invention, Wold's double pointer to a static function 531 fails to disclose the claimed variable.

In addition to the foregoing distinctions between the claimed invention and Wold, the Applicant respectfully submits that Wold's "closure" (i.e., a double pointer in a calling object to a static dispatcher member function in a called object in order to invoke a virtual function in the called object) further does not disclose a software unit which comprises: an output gate for transmitting a message to invoke a method at one or more of the plurality of software units; and a variable accessed at runtime of the software unit that identifies the one or more of the plurality of software units that is to receive the message and identifies the method that is to be executed at the identified one or more of the plurality of software units, as particularly recited in the independent Claims 27 and 47. In contrast to the presently claimed software unit's variable, Wold's closure requires a cumbersome syntax. Further in contrast to the claimed software unit's output gate, Wold requires an explicit representation for every "closure" used by a calling object, i.e., calling object of Class B 530. Therefore, the type of reuse provided by Wold's "closure" is completely distinct from the claimed invention and is further extremely limited as compared to the claimed invention as follows.

The Applicant provides the following illustrative examples to illustrate the differences between the software unit of the claimed invention and the Wold's "closure" concept. Thus, the first example is of a claimed exemplary software unit, e.g., a "Button" software unit:

Button::outGates

```
        ^super outGates add: #clicked
Button::onClick
        out clicked.
```

In the foregoing software unit example, the "Button" is able to send a message "clicked" (indicating that the Button has been clicked by a user) to the output gate "#clicked" when the Button receives a message "onClick." The variable "out" identifies the one or more software units that is to receive the message sent to the output gate. The variable out further identifies the method that is to be executed at the identified one or more of the plurality of software units at run time, set by a linkage mechanism that establishes the connections from the Button's output gate #clicked to the identified one or more of the plurality of software units. The claimed software unit, as exemplified by the exemplary Button, is therefore reusable because it does not depend on the number of the plurality of software units that may receive the message from the software unit.

In contrast to the claimed exemplary software unit Button, the second example is of a Button that is enabled to call three (3) closures using Wold's "closure" concept, implemented in Smalltalk syntax:

```
WoldButton3
        instanceVariables: 'closure1 closure2 closure3'
WoldButton3::setClosure1: aClosure
        closure1 := aClosure
WoldButton3::setClosure2: aClosure
        closure2:= aClosure
WoldButton3::setClosure3: aClosure
        closure3 := aClosure
WoldButton::onClick
        closure1 value.
        closure2 value.
        closure3 value.
```

In the foregoing Wold "closure" concept, a new instance variable (closure1, closure2, closure3) must be created in the WoldButton3 object to store each closure, i.e., for each

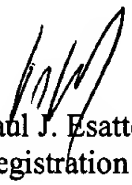
object that is to receive a message from the WoldButton3. The Wold's method "onClick" must explicitly invoke each closure by sending a method "value" to the closures (closure1, closure2, closure3). Thus, the Wold's method "onClick" in the WoldButton3 object is limited to three (3) closures and a new WoldButton object needs to be created to handle, for example, five (5) closures. More specifically, a new WoldButton object would need to be created for any different number of closures, as particularly required. As exemplified by the WoldButton3, Wold's reusability is limited because the definition of WoldButton3 object would need to be modified for the different number of closures required.

Consequently, in contrast to the Wold's "closure" concept that requires a closure instance variable to be created for each object (within the WoldButton3 object) that is to receive a message from the WoldButton3 object (thereby requiring modification of WoldButton3 for the different number of objects that receive the message), the claimed exemplary software unit Button can transmit the message to an arbitrary number of software units (i.e., one or more software units) without any modification of the Button software unit. Therefore, the claimed invention vastly improves reusability over reusability that can be obtained by Wold.

In view of the foregoing, the Applicant respectfully requests the Examiner to withdraw the rejection of the independent Claims 27 and 47 pursuant to 35 U.S.C. § 102(b) and allow these claims. The Applicant further respectfully requests the Examiner to allow the dependent Claims 28-46 based at least on their respective dependencies, whether direct or indirect, from the independent Claim 27.

In view of the foregoing, the Applicant believes that the above-identified application is in condition for allowance and henceforth respectfully solicits the allowance of the application. If the Examiner believes a telephone conference might expedite the allowance of this application, the Applicant respectfully requests that the Examiner call the undersigned, Applicant's attorney, at the following telephone number: (516) 742-4343.

Respectfully submitted,



Paul J. Esatto, Jr.
Registration No. 30,749

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343
AGV:eg/yd